```
PPPPPPPPPPPP     LLL                  IIIIIIIII     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
PPPPPPPPPPPP     LLL                  IIIIIIIII     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
PPPPPPPPPPPP     LLL                  IIIIIIIII     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
PPP         PPP  LLL                     III        RRR         RRR        TTT        LLL
PPP         PPP  LLL                     III        RRR         RRR        TTT        LLL
PPP         PPP  LLL                     III        RRR         RRR        TTT        LLL
PPP         PPP  LLL                     III        RRR         RRR        TTT        LLL
PPP         PPP  LLL                     III        RRR         RRR        TTT        LLL
PPPPPPPPPPPP     LLL                     III        RRRRRRRRRRRR           TTT        LLL
PPPPPPPPPPPP     LLL                     III        RRRRRRRRRRRR           TTT        LLL
PPPPPPPPPPPP     LLL                     III        RRRRRRRRRRRR           TTT        LLL
PPP              LLL                     III        RRR   RRR              TTT        LLL
PPP              LLL                     III        RRR   RRR              TTT        LLL
PPP              LLL                     III        RRR   RRR              TTT        LLL
PPP              LLL                     III        RRR      RRR           TTT        LLL
PPP              LLL                     III        RRR      RRR           TTT        LLL
PPP              LLL                     III        RRR      RRR           TTT        LLL
PPP              LLLLLLLLLLLLLLL      IIIIIIIII     RRR         RRR        TTT        LLLLLLLLLLLLLLL
PPP              LLLLLLLLLLLLLLL      IIIIIIIII     RRR         RRR        TTT        LLLLLLLLLLLLLLL
PPP              LLLLLLLLLLLLLLL      IIIIIIIII     RRR         RRR        TTT        LLLLLLLLLLLLLLL
```

```
_$2

Sym
---
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI

PLI
PLI
PLI

PLI
PLI
PLI
PLI
PLI
PLI
PLI

PLI
PLI
PLI
PLI

PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI
```

```
**FILE**ID**PLIHEEP

PPPPPPPP   LL          IIIIII   HH      HH  EEEEEEEEEE  EEEEEEEEEE  PPPPPPPP
PPPPPPPP   LL          IIIIII   HH      HH  EEEEEEEEEE  EEEEEEEEEE  PPPPPPPP
PP      PP LL            II     HH      HH  EE          EE          PP      PP
PP      PP LL            II     HH      HH  EE          EE          PP      PP
PP      PP LL            II     HH      HH  EE          EE          PP      PP
PP      PP LL            II     HH      HH  EE          EE          PP      PP
PPPPPPPP   LL            II     HHHHHHHHHH  EEEEEEEE    EEEEEEEE    PPPPPPPP
PPPPPPPP   LL            II     HHHHHHHHHH  EEEEEEEE    EEEEEEEE    PPPPPPPP
PP         LL            II     HH      HH  EE          EE          PP
PP         LL            II     HH      HH  EE          EE          PP
PP         LL            II     HH      HH  EE          EE          PP          ....
PP         LL            II     HH      HH  EE          EE          PP          ....
PP         LLLLLLLLLL  IIIIII   HH      HH  EEEEEEEEEE  EEEEEEEEEE  PP          ....
PP         LLLLLLLLLL  IIIIII   HH      HH  EEEEEEEEEE  EEEEEEEEEE  PP          ....

LL          IIIIII      SSSSSSSS
LL          IIIIII      SSSSSSSS
LL            II      SS
LL            II      SS
LL            II      SS
LL            II      SS
LL            II        SSSSSS
LL            II        SSSSSS
LL            II              SS
LL            II              SS
LL            II              SS
LL            II              SS
LLLLLLLLLL  IIIIII    SSSSSSSS
LLLLLLLLLL  IIIIII    SSSSSSSS
```

Table of contents

PL

```
0000           .title pli$heep - pl1 runtime heep allocation
0000           .ident  /1-003/                                 ; Edit WHM1003
0000   ;
0000   ;****************************************************************************
0000   ;*                                                                          *
0000   ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000   ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000   ;*  ALL RIGHTS RESERVED.                                                    *
0000   ;*                                                                          *
0000   ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000   ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000   ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000   ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000   ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000   ;*  TRANSFERRED.                                                            *
0000   ;*                                                                          *
0000   ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000   ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000   ;*  CORPORATION.                                                            *
0000   ;*                                                                          *
0000   ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000   ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000   ;*                                                                          *
0000   ;*                                                                          *
0000   ;****************************************************************************
0000   ;
0000   ;++
0000   ; facility:
0000   ;
0000   ;       VAX/VMS PL1 runtime library.
0000   ;
0000   ; abstract:
0000   ;
0000   ;       Runtime routines to allocate and deallocate heep space.
0000   ;
0000   ; author: R. Heinen  16-dec-1978
0000   ;
0000   ; modifications:
0000   ;
0000   ;       1-002
0000   ;       Modified both routines, to allow them be called with different
0000   ;       number of parameters and so work for the controlled variable.
0000   ;                               Alex Wu  03/24/82
0000   ;
0000   ;       add the new routine - PLI$ALLOCATION to return the number of
0000   ;       generation that a controlled variable has been allocated.
0000   ;                               Alex Wu  03/30/82
0000   ;
0000   ;       1-003   Bill Matthews   29-September-1982
0000   ;
0000   ;               Invoke macros $defdat and rtshare instead of $defopr and share.
0000   ;
0000
0000   ;
0000   ; external definitions
0000   ;
0000
```

```
                             0000     ;
                             0000     ; local data
                             0000     ;
                             0000
                             0000             rtshare
```

```
                               0000     64          .sbttl pli$aloheep - allocate heep space
                               0000     65  ;++
                               0000     66  ; pli$alocheep - allocate heep space
                               0000     67  ;
                               0000     68  ; functional description:
                               0000     69  ;
                               0000     70  ; This routine allocates a memory block using "lib$get_vm".
                               0000     71  ; The alloocated block is a longword (or two longword) bigger in order to
                               0000     72  ; save the allocated size (and/or the previous pointer)in the block itself.
                               0000     73  ; The size is stored in the first longword (the pointer is save in the first
                               0000     74  ; longword).  The returned address
                               0000     75  ; is the address of the actual free space.
                               0000     76  ;
                               0000     77  ; inputs:
                               0000     78  ;
                               0000     79  ;       (ap) = 2 (or 3 for the controlled variables)
                               0000     80  ;       4(ap) = longword size
                               0000     81  ;       8(ap) = address to store address of the data
                               0000     82  ;       12(ap) = address to the current block -- if exist
                               0000     83  ;
                               0000     84  ; outputs:
                               0000     85  ;
                               0000     86  ;       r0 = success indicator fixed binary(32)
                               0000     87  ;
                               0000     88  ; r0 =  ss$_normal for success
                               0000     89  ;       lib$_insvirmem - insufficient memory for request
                               0000     90  ;       lib$_badblosiz - bad size parameter
                               0000     91  ;
                               0000     92  ; If an error is indicated by r0 then the returned address is 0.
                               0000     93  ; and the error is signalled.
                               0000     94  ;--
                         0000  0000     95          .entry pli$alocheep,0
                               0002     96
           08 BC         D4    0002     97          clrl    @8(ap)                  ; assume allocation failure
   7E   04 AC    04      C1    0005     98          addl3   #4,4(ap),-(sp)          ; build following arg list
           6C    02      D1    000A     99          cmpl    #2,(ap)                 ; based ?
                 03      13    000D    100          beql    10$                     ; less than or equal then yes
           6E    04      C0    000F    101          addl2   #4,(sp)                 ; one more longword for pointer
                 7E      D4    0012    102 10$:     clrl    -(sp)                   ; address to return address
                 6E      9F    0014    103          pushab  (sp)                    ;
              08 AE      9F    0016    104          pushab  8(sp)                   ;
   00000000'GF   02      FB    0019    105          calls   #2,g^lib$get_vm         ; allocate the memory
           14    50      E9    0020    106          blbc    r0,20$                  ; if low clear then error
           50  8ED0            0023    107          popl    r0                      ; get address of allocated space
           80  8ED0            0026    108          popl    (r0)+                   ; insert size in buffer
           6C    02      D1    0029    109          cmpl    #2,(ap)                 ; based ?
                 04      13    002C    110          beql    15$                     ; less than or equal than yes
        80    0C AC      D0    002E    111          movl    12(ap),(r0)+            ; save pointer
        08 BC    50      D0    0032    112 15$:     movl    r0,@8(ap)               ; return address of allocated memory
                         04    0036    113          ret
                 50      DD    0037    114 20$:     pushl   r0                      ; signal error condition
   00000000'GF   01      FB    0039    115          calls   #1,g^lib$signal
                         04    0040    116          ret
```

```
                                    0041          .sbttl pli$freeheep - deallocate heep space
                                    0041     ;++
                                    0041     ; pli$freeheep - deallocate heep space
                                    0041     ;
                                    0041     ; functional description:
                                    0041     ;
                                    0041     ; This routine is the complementary routine to "pli$alocheep".
                                    0041     ; The memory is retured via "lib$free_vm".
                                    0041     ;
                                    0041     ; inputs:
                                    0041     ;
                                    0041     ;         (ap) = 1 (or 2 for the controlled variables)
                                    0041     ;        4(ap) = address of memory
                                    0041     ;        8(ap) = address of the based pointer -- if exist
                                    0041     ;
                                    0041     ; The block must have been allocated using "pli$aloheep".
                                    0041     ;
                                    0041     ; outputs:
                                    0041     ;
                                    0041     ;         For the controlled variables, there is a side effect on the
                                    0041     ;         based pointer @8(ap)
                                    0041     ;         r0 = success indicator fixed binary(32)
                                    0041     ;
                                    0041     ; r0 =   ss$_normal for success
                                    0041     ;        lib$_badblosiz - incorrect block size parameter
                                    0041     ;--
                              0000  0041              .entry pli$freeheep,0
                                    0043
     50    04 AC    04    C3  0043              subl3   #4,4(ap),r0            ; address block size or pointer longword
              6C    01    D1  0048              cmpl    #1,(ap)                ; based variable ?
                    07    13  004B              beql    30$                    ; less than or equal than yes
           08 BC    60    D0  004D              movl    (r0),@8(ap)            ; save previous block pointer
              50    04    C2  0051              subl2   #4,r0                  ; adress to block size
                    50    DD  0054     30$:     pushl   r0                     ; put address in memory
                    6E    9F  0056              pushab  (sp)                   ; set up arg list
                    50    DD  0058              pushl   r0                     ;
  00000000'GF       02    FB  005A              calls   #2,g^lib$free_vm       ; free the memory
                          04  0061              ret
```

```
                             0062           .sbttl pli$allocation - return number of time that has been allocated
                             0062      ;++
                             0062      ; pli$allocation - return number of time that has been allocated
                             0062      ;
                             0062      ; functional description:
                             0062      ;
                             0062      ; This routine perform the builtin funtion ALLOCATION which return the
                             0062      ; generations that a controlled variable has been allocated.  It walks
                             0062      ; through the link list and increment the count each time looks at a
                             0062      ; new block until the end of the list (null pointer).
                             0062      ;
                             0062      ; inputs:
                             0062      ;
                             0062      ;        (ap) = 2
                             0062      ;        4(ap) = address of the current active block
                             0062      ;        8(ap) = address of the return value
                             0062      ;
                             0062      ; The block must have been allocated using "pli$aloheep".
                             0062      ;
                             0062      ; outputs:
                             0062      ;
                             0062      ;        8(ap) contains the return value
                             0062      ;--
                      0000   0062           .entry pli$allocation,0
                             0064
            08 BC     D4     0064           clrl    @8(ap)                  ; initialize the counter
     50     04 AC     D0     0067           movl    4(ap),r0                ; get the pointer
               08     13     006B           beql    30$                     ; if null pointer then done
            08 BC     D6     006D      10$: incl    @8(ap)                  ; increment the counter
        50     70     D0     0070           movl    -(r0),r0                ; get previous block address
               F8     12     0073           bnequ   10$                     ; keep looping if not null pointer
                      04     0075      30$: ret
                             0076
                             0076           .end
```

```
LIB$FREE_VM      ********   X    01
LIB$GET_VM       ********   X    01
LIB$SIGNAL       ********   X    01
PLI$ALLOCATION   00000062 RG     01
PLI$ALOCHEEP     00000000 RG     01
PLI$FREEHEEP     00000041 RG     01

                                              +-----------------+
                                              ! Psect synopsis !
                                              +-----------------+

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (     0.)  00 (  0.)  NOPIC    USR    CON    ABS    LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
_PLI$CODE                       00000076  (   118.)  01 (  1.)    PIC    USR    CON    REL    LCL   SHR   EXE   RD   NOWRT NOVEC LONG

                                           +--------------------------+
                                           ! Performance indicators !
                                           +--------------------------+

Phase                        Page faults    CPU Time       Elapsed Time
-----                        -----------    --------       ------------
Initialization                         9    00:00:00.05    00:00:00.59
Command processing                    68    00:00:00.54    00:00:01.51
Pass 1                                61    00:00:00.55    00:00:01.26
Symbol table sort                      0    00:00:00.00    00:00:00.00
Pass 2                                43    00:00:00.38    00:00:00.60
Symbol table output                    1    00:00:00.01    00:00:00.01
Psect synopsis output                  1    00:00:00.02    00:00:00.02
Cross-reference output                 0    00:00:00.00    00:00:00.00
Assembler run totals                 183    00:00:01.56    00:00:04.00

The working set limit was 750 pages.
2560 bytes (5 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 6 non-local and 6 local symbols.
190 source lines were read in Pass 1, producing 17 object records in Pass 2.
1 page of virtual memory was used to define 1 macro.

                                          +----------------------------+
                                          ! Macro library statistics !
                                          +----------------------------+

Macro library name                        Macros defined
------------------                        --------------
_$255$DUA28:[PLIRTL.OBJ]PLIRTMAC.MLB;1                 1
_$255$DUA28:[SYSLIB]STARLET.MLB;2                      0
TOTALS (all libraries)                                 1

3 GETS were required to define 1 macros.

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE=TRACEBACK/LIS=LIS$:PLIHEEP/OBJ=OBJ$:PLIHEEP MSRC$:PLIHEEP/UPDATE=(ENH$:PLIHEEP)+LIB$:PLIRTMAC/LIB
```

0308 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

PLIFORMAT
LIS

PLIGETBUF
LIS

PLIMSGTXT
LIS

PLIGETEDI
LIS

PLIHEEP
LIS

PLIPUTFIL
LIS

PLIRMSBIS
LIS

PLIRECOPT
LIS

PLIOPEN
LIS

PLIREAD
LIS

PLIPROTEC
LIS

PLIREWRIT
LIS

PLIGETLIS
LIS

PLIPUTEDI
LIS

PLIPKDIUL
LIS

PLIPUTLIS
LIS

PLIMSGPTR
LIS

PLIPKDIUS
LIS

PLIPUTBUF
LIS

PLIGETFIL
LIS